Équipe A* (208)

Poly Diff
Plan de projet

Version 1.8

Historique des révisions

Date	Version	Description	Auteur
2024-01-17	1.0	Rédaction de l'introduction	Jeremy Rouillard
2024-01-20	1.1	Rédaction solution proposée	Jeremy Rouillard
2024-01-20	1.2	Rédaction de la gestion des exigences	Ryan Lahbabi
2024-01-24	1.3	Rédaction du contrôle de la qualité	Ryan Lahbabi
2024-01-27	1.4	Rédaction de la gestion de la configuration	Ryan Lahbabi
2024-01-28	1.5	Rédaction de l'équipe de développement	Ryan Lahbabi
2024-01-28	1.6	Rédaction de l'entente contractuelle proposée	Ryan Lahbabi
2024-01-28	1.7	Révision du contenu du rapport	Jeremy Rouillard
2024-01-30	1.8	Rédaction de l'échéancier du projet	Ryan Lahbabi

Table des matières

1. Introduction **4**
2. Énoncé des travaux **4**
2.1. Solution proposée 4
2.2. Hypothèses et contraintes 4
2.3. Biens livrables du projet 5
3. Gestion et suivi de l'avancement **6**
3.1. Gestion des exigences 6
3.2. Contrôle de la qualité 7
3.3. Gestion de risque 9
3.4. Gestion de configuration 11
4. Échéancier du projet **12**
5. Équipe de développement **16**
6. Entente contractuelle proposée **17**

Plan de projet

1. Introduction

En abordant notre dernière année à Polytechnique, nous avons eu l'opportunité de collaborer avec une diversité de personnes, une expérience enrichissante qui a mené à l'élaboration d'un produit. Ce dernier représente une continuité du travail entrepris dans le cadre du cours LOG2990, projet de logiciel d'application Web. Le plan de notre projet, qui fera l'objet de ce rapport, se divise en plusieurs sections essentielles : les déclarations de travaux, la gestion et le suivi des progrès, le calendrier du projet, la composition de l'équipe de développement, ainsi que les termes de l'accord contractuel. Cette introduction vise à donner un aperçu du contexte et de la structure de notre projet.

2. Énoncé des travaux

2.1. Solution proposée

Ici la solution proposée par l'équipe est l'application Poly Diff. Cette application est en gros un jeu en ligne dans lequel les gens doivent trouver des différences entre deux photos. La solution est composée de plusieurs modules tels que le client lourd, le client léger et le serveur. Le client lourd a déjà été implémenté pour la majeure partie puisqu'elle avait été programmée en Typescript dans le cadre du cours LOG2990, projet de logiciel d'application Web. Cependant, le client léger n'a pas encore débuté en ce début de projet. Celui-ci sera conceptualisé à l'aide du langage de programmation dart et le cadre de travail Flutter. Toutes les communications du client lourd bénéficiant d'une communication bi-directionnelle ont été faites à l'aide de Sockets. Autrement, ce sont des requêtes HTTP qui ont été utilisées

2.2. Hypothèses et contraintes

Dans le plan du projet, nous avons tout d'abord un projet existant dont le code est très flexible face à l'ajout de nouvelles fonctionnalités. Nous avons cependant quelques contraintes. Nous avons une petite équipe pour réaliser le produit. Il faudra ainsi trouver les forces de chacun dans l'équipe afin de maximiser les résultats avec les ressources à notre disposition. Tout le monde dans l'équipe possède un ordinateur suffisamment puissant pour rouler les applications que nous allons développer. Toutefois, l'équipe n'a qu'une seule tablette Samsung Galaxy S6 lite. Par conséquent, il faudra trouver un moyen de gérer le partage de la tablette pour ceux qui développeront les fonctionnalités sur l'application mobile. Dans notre projet, nous avons défini un calendrier qui comprend des échéances spécifiques pour les livrables, comme mentionné dans la section 2.3. Nous avons également planifié trois sprints de développement de durées différentes.

Le premier sprint, qui dure un mois, est axé sur le développement des fonctionnalités du serveur, l'amélioration du client léger, et l'ajout d'un premier niveau de fonctionnalités qui répond aux exigences initiales que nous nous sommes fixées.

Pour notre deuxième sprint, d'une durée de deux semaines, nous nous concentrerons sur la poursuite du déploiement de nouvelles fonctionnalités. Étant donné que la majeure partie du travail aura été accomplie lors du premier sprint, nos efforts seront axés sur des améliorations itératives. Pendant ce sprint, il sera essentiel pour chaque membre de l'équipe de garder à l'esprit les tâches futures et les implications à long terme de leur travail actuel.

Le troisième et dernier sprint, également d'une durée de deux semaines, sera consacré à la finalisation du projet. Notre objectif principal sera de garantir la qualité de l'application. Cette phase sera cruciale pour peaufiner les détails, éliminer les bugs éventuels et s'assurer que l'application fonctionne de manière optimale avant son lancement.

Nous avons également décidé de centrer notre travail sur les fonctionnalités plutôt que sur la répartition des tâches par section. Cette approche nous oblige tous à travailler en mode full stack. Bien que cela représente un défi, l'avantage est que chaque membre de l'équipe sera capable de contribuer à toutes les sections du projet, éliminant ainsi les risques de blocage. Cette stratégie vise à renforcer la collaboration et l'efficacité de notre équipe dans son ensemble.

2.3. Biens livrables du projet

Afin de développer un produit concis. Nous nous devons de rédiger des artefacts pour l'appel d'offres. De cette information, nous devons développer les documents suivants.

Document à rédiger	**Date de livraison**
Spécifications des requis du système (SRS)	24 janvier 2024
Protocole de communication	9 février 2024
Architecture logicielle	9 février 2024
Plan de tests logiciels	15 avril 2024
Résultats de tests logiciels	15 avril 2024
Plan de projet	9 février 2024
Mettre à jour : - Spécifications des requis du système (SRS) - Protocole de communication - Architecture logicielle - Plan de projet	15 avril 2024

Tableau 1. Dates de livraison associés au document à rédiger

3. Gestion et suivi de l’avancement

3.1. Gestion des exigences

Pour le suivi des exigences, nous utiliserons la plateforme JIRA. Des tickets seront émis pour chaque tâche et attribués à divers membres de l’équipe. Le niveau de priorité sera défini ainsi que le temps estimé pour que la tâche soit réalisée. Chaque exigence sera liée à ce que l’on appelle un epic dans Jira, pour définir une catégories de tâches et surtout pour diviser plusieurs une tâche en plusieurs sous-tâches.

Les tâches hebdomadaires seront sélectionnées en équipes lors de notre rencontre en présentiel et réparties parmi les différents membres de façon équitable. Elles seront ensuite déplacées du backlog au sprint actif dans la colonne “À faire”. Dès que le développeur commence cette tâche, il la déplace dans la colonne “en cours” et enregistre le temps consacré à cette tâche à chaque fois qu’il travaille dessus sur la dite tâche. Cela sera accompagné de commentaires afin de décrire ce qu’il a fait pendant la période de travail en question. Enfin, une tâche qui sera terminée sera déplacée dans la colonne “terminée” de notre tableau Kanban.

Afin de maintenir la clarté et la pertinence des exigences, une revue régulière sera effectuée. L'équipe discutera des exigences au fur et à mesure du projet. Toute modification nécessaire doit être approuvée par l'équipe avant son ajout. En cas de modification d’une exigence, toutes les exigences potentiellement affectées seront examinées pour assurer la cohérence et éviter les problèmes. En général, nous ferons en sorte que les modifications soient choisies pour minimiser l’impact sur les autres exigences. Nos réunions en présentiel hebdomadaire et notre Scrum du samedi nous permettront de suivre l’avancement de ces exigences.

3.2. Contrôle de la qualité

Date	Moment de contrôle	Méthodes utilisées
10 Janvier	Début du projet: Rencontre initiale	- Établir un inventaire exhaustif de tous les produits et documents à fournir. - Examiner globalement tous les documents afin que l'équipe dispose d'une compréhension globale des tâches à accomplir.
17 Janvier	1er point de contrôle: Une fois validation des exigences des chargés	- Effectuer une première évaluation de notre progression à l'aide de JIRA et se mettre d'accord sur notre organisation sur la plateforme. - Programmer les prochaines tâches par rapport aux exigences sélectionnées.
24 Janvier	2e point de contrôle: Avant la remise du SRS pour le cours de LOG3000	Vérification du SRS en équipe afin de vérifier nos exigences fonctionnelles et non fonctionnelles
1 à 2 semaines après le 25 Janvier	3e point de contrôle: Suite à la correction du SRS	Mettre à jour les documents en intégrant les retours sur notre travail pratique dans l'objectif d'augmenter la qualité du SRS.
3 Février	4e point de contrôle: Révision avant la réponse à l'appel d'offres	- Révision du code du prototype par les membres de l'équipe. - Vérification du bon fonctionnement du logiciel par rapport aux exigences demandées. - Révision de la qualité des différents documents à remettre et de la cohérence entre les diagrammes, nos exigences et l'architecture de notre projet.
1 à 2 semaines après le 9 février	5e point de contrôle: Révision après la correction de l'appel d'offre	Ajouter des tickets JIRA suite aux suggestions des correcteurs afin d'améliorer notre documentation.
4 Mars	6e point de contrôle: avant la fin du sprint 1	- Analyse des fonctionnalités et identification des anomalies. - Ajout de ticket JIRA en conséquence.
18 Mars	7e point de contrôle avant la fin du sprint 2	- Analyse des fonctionnalités et identification des anomalies. - Ajout de ticket JIRA en conséquence.
3 Avril	8e point de contrôle: avant sprint 3 et la remise du Produit final	- Évaluation de l'expérience utilisateur : inviter des utilisateurs qui ne font pas partie du projet à essayer l'application sur des appareils mobiles et des ordinateurs, recueillir leurs retours pour perfectionner le produit. - Analyse des fonctionnalités et identification des anomalies.

Tableau 2. Point de contrôle relié à la gestion

En plus de ces moments ponctuel du contrôle de la qualité des biens livrables du projet, nous aurons plusieurs pratiques régulières durant les prochaines semaines qui aideront à entreprendre des actions correctives lorsque nécessaire:

- Révision des merge requests par au minimum deux membres de l'équipe.
- Nos deux Scrum hebdomadaire nous permettent de faire des contrôles réguliers du travail de chacun.
- Des tests logiciels et d'utilisabilité seront effectués afin de détecter les défauts potentiels pendant le développement de notre application.
- Des tâches supplémentaires seront créées sur la plateforme JIRA afin de corriger les défauts détectés et seront traités sur une branche à part. Une fois le défaut corrigé, le développeur concerné fera une merge request qui sera vérifiée par deux autres membres de l'équipe.
- Une révision des tickets Jira afin d'assurer la bonne gestion des tâches.

3.3. Gestion de risque

La description des risques suit la convention suivante :

- Ampleur : sur une échelle de 1 à 10, 10 étant le risque le plus élevé. Cette analyse est basée sur la probabilité d'occurrence du risque, ainsi que ses impacts.
- Description : une description textuelle du risque ainsi que les problèmes attendus.
- Impact : échelle définissant la portée du risque
    - C – critique (affecte le projet en entier)
    - E – élevé (affecte les fonctionnalités principales du système)
    - M – moyen (devrait être maîtrisable en appliquant une stratégie d'atténuation adéquate)
    - F – faible (l'acceptation du risque est une stratégie envisageable)
- Facteurs : aspects (**métriques**) du système pouvant être compromis.
- Stratégie de gestion : mesures à prendre afin de gérer le risque.

1 - Minimiser une exigence nécessaire				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Une exigence nécessaire a été sous estimée et exige plus de temps et de ressources que prévus	E	-Ressources humaines -Délais de livraison	Nous allons nous focaliser davantage dessus et travailler à plusieurs membres sur cette exigence. Changer la priorité du ticket JIRA en élevée afin de rattraper le temps sous estimé.

2 - Minimiser une exigence souhaitable				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Une exigence souhaitable a été sous estimée et exige plus de temps et de ressources que prévus	F	-Ressources humaines -Délais de livraison -Charge de travail	Changer d'exigence souhaitable entre les différents choix que nous avons; s'il n'y a pas plus d'exigences disponibles, nous allons nous focaliser davantage dessus et travailler à plusieurs membres sur cette exigence.

3 - Membres de l'équipe non disponible				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Un ou plusieurs membres de l'équipe n'est pas disponible pendant une ou plusieurs semaines pour différentes raisons(maladie, problème familial etc.)	M	-Ressources humaines -Délais de livraison -Charge de travail	Se répartir les tâches entre les différents membres de l'équipe afin de pallier à l'absence du coéquipier et changer la responsabilité de ses tâches au niveau des tickets JIRA.

4 -Défaillance dans la sauvegarde de données				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	Risque de perte de données importantes en raison de défaillances dans les systèmes de sauvegarde, incluant des défaillances techniques ou des problèmes logiciels.	C	-Intégrité des donnée -Fiabilité de la base de donnée	Mettre en place une stratégie de sauvegarde robuste, incluant des sauvegardes régulières et l'utilisation de solutions de sauvegarde redondantes

5 - Bogue dans l'interface utilisateur				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
8	L'utilisateur rencontre des problèmes à exécuter certaines tâches	E	Application non testée de façon exhaustive	Faire des tests utilisateurs régulièrement pour vérifier que tout marche bien, essayer tous les cas d'utilisations. Si un problème est rencontré par la suite, créez un ticket JIRA pour le régler.

6 - Fluctuation de performance du serveur				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Le serveur est moins stable lorsque beaucoup d'utilisateur sont connectés au même moment	E	-Performance de nos instances -Complexité algorithmique élevée	Améliorer la complexité de nos algorithmes au niveau du serveur, prendre une instance EC2 AWS plus puissante.

7 - Interface utilisateur non intuitive				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Certaines fonctionnalités ne sont pas assez intuitives pour l'utilisateur que ce soit l'accès ou l'utilisation de cette dernière.	M	-Expérience utilisateur non consultée auprès d'utilisateurs aléatoires - Figma mal conçu	Revoir l'UX sur Figma, l'implémenter et demander à une personne hors de l'équipe de tester le changement pour vérifier qu' une fois que le changement est établi, l'application est plus intuitive.

3.4. Gestion de configuration

Pour la gestion de notre projet, nous employons l'outil JIRA, qui nous permet de suivre les bugs et de gérer les opérations du projet. Après nos réunions hebdomadaires d'équipe, nous y inscrivons les tâches hebdomadaires à réaliser, et chaque membre y consignera le temps passé sur chaque tâche. Les tickets auront une description de la tâche avec la date pour laquelle il faut l'avoir terminé. Chacun sera intégré à un epic, qui regroupe plusieurs sous-tâches liés à une plus grande tâche, afin d'organiser nos tickets efficacement. Chaque ticket inclura un titre explicite et une description de la tâche demandée, ainsi que le nombre d'heures estimées pour sa réalisation. Les bugs sont traités comme des tâches standard, ils auront une étiquette "Bogue" sur leur ticket JIRA.

En parallèle, Gitlab est utilisé pour la gestion de version du code source. Nous y créerons différentes branches pour de nouvelles fonctionnalités et les fusionnerons avec la branche principale une fois complétées. La branche main contient la version de l'application prête pour la production, tandis que des branches spécifiques sont utilisées pour le développement de fonctionnalités et la correction de bugs. Le ticket JIRA concerné sera toujours spécifié. Voici la nomenclature de nos branches:

- "mobile/feature/{numero-ticket-Jira}_{nom-de-la-feature}"
- "desktop/bogue/{numero-ticket-Jira}_{nom-du-bogue}"
- "server/feature/{numero-ticket-Jira}_{nom-de-la-feature}"

Après avoir terminé une tâche, le développeur responsable doit soumettre une demande de fusion (merge request) pour intégrer sa branche à la branche main. Cette demande doit inclure une description générale de ses contributions et de la tâche. Deux autres développeurs examineront cette demande et donneront leur avis. Le développeur responsable devra ajuster son code en fonction de ces retours pour obtenir leur approbation. Une fois la demande approuvée, la branche peut être fusionnée avec la branche main.

4. Échéancier du projet

Nous avons 14 semaines afin de réaliser notre application mobile et notre application web avec leur documentation respective. Sachant que chaque développeur et gestionnaire doit travailler 180 heures sur le projet, cela fait un total de 1080 heures*personnes pour le projet au total. Cela revient à 77 heures*personnes par semaine de travail pour toute l'équipe. Nous allons alors répartir ces heures en fonction de la durée de chacun de nos 4 sprints afin d'être capable de développer toutes les fonctionnalités du projet demandé.

Sprint 1 (4 semaines): 77*4 = 308 heures*personnes

Sprint 2 (3 semaines): 77*3 = 231 heures*personnes

Sprint 3 (2 semaines): 77*2 = 144 heures*personnes

Sprint 4 (2 semaines): 77*2 = 144 heures*personnes

Les heures par lots ont été choisies par rapport à la valeur en point de la fonctionnalité, l'estimation que nous avons jugé que cela prendra à développer et le sprint où nous comptons développer le lot. En effet, nous considérons qu'une fonctionnalité ayant un même nombre de points qu'un autre sera plus rapide à développer au sprint 3 qu' au sprint 1, de par l'expertise en programmation que nous aurons acquis au cours des semaines. Nous considérons que nos rencontres en présentiel hebdomadaire afin de discuter sur l'organisation du projet et de planifier les tâches de la semaine durent 2h donc un temps*personne hebdomadaire de 12h pour nous 6. Quant au Scrum hebdomadaire, il dure 15 minutes donc un temps*personnes de 1h30 hebdomadaire.

Sprints	***Date de début - Date de fin***	***Principaux lots de travail***	***Effort estimé***	
			Nombre d'heures par lots	*Total du sprint*
Sprint 0 (4 semaines)	*17/01/2024 - 14/02/2024*	SRS	35h	**308h**
		Plan de projet	18h	
		Protocole de communication	23h	
		Document d'architecture logicielle	30h	
		Formation sur Flutter	24h	
		Prototype du client léger	40h	
		Prototype du client lourd	40h	
		Choix et négociation de la liste d'exigences	14h	
		Rencontre hebdomadaire en équipe et Scrum à distance	54h	
		Mettre en place la pipeline du projet sur un serveur rapide	10h	
		Conception Figma des nouvelles pages à ajouter	15h	
		Organisation initiale de Jira	5h	

Sprint 1 (3 semaines)	*15/02/2024 - 06/03/2024*	**Client lourd:** Exigences Essentielles	Clavardage - Intégration (2 points)	5h	**231h** **76 pts**
			Clavardage - canaux de discussion (1 point)	3h	
			Compte utilisateur et historique (3 points)	7h	
			Avatar (1 point)	3h	
			Modes de jeu (12 points)	30h	
			Vue d'administration et vue de création (6 points)	14h	
			Observateurs (6 points)	14h	
			Personnalisation de l'application (2 points)	4h	
			Système d'amis (7 points)	16h	
		Client léger: Exigences Essentielles	Clavardage - Intégration (4 points)	12h	
			Clavardage - canaux de discussion (1 point)	3h	
			Compte utilisateur et historique (2 points)	6h	
			Avatar (1 point)	3h	
			Modes de jeu (10 points)	28h	
			Clavardage - messages globaux (1 point)	3h	
			Clavardage - messages aléatoires (1 point)	3h	
			Clavardage - Envoi des GIF (2 points)	5h	
			Profil utilisateur (1 point)	3h	
			Observateurs (6 points)	14h	
			Personnalisation de l'application (2 points)	5h	
			Système d'amis (5 points)	12h	
		Rencontre hebdomadaire en présentiel et Scrum à distance		38h	
Sprint 2 (2 semaines)	*07/03/2024 - 20/03-/2024*	**Client lourd:** Exigences Essentielles	Clavardage - Intégration (+ 3 points)	8h	**144h** **49 pts**
			Clavardage - canaux de discussion (+ 2 points)	4h	
			Compte utilisateur et historique (+ 2 points)	6h	
			Avatar (+ 1 point)	3h	
			Modes de jeu (+ 4 points)	8h	

			Vue d'administration et vue de création (+ 2 points)	6h	
			Observateurs (+ 4 points)	10h	
			Personnalisation de l'application (4 points)	10h	
			Reprise vidéo (6 points)	12h	
		Client léger: Exigences Essentielles	Clavardage - canaux de discussion (+ 2 points)	4h	
			Compte utilisateur et historique (+ 1 point)	4h	
			Avatar (+ 1 point)	2h	
			Observateurs (+ 2 points)	8h	
			Personnalisation de l'application (4 points)	10h	
			Système d'amis (+ 4 points)	10h	
			Reprise vidéo (5 points)	12h	
		Rencontre hebdomadaire en présentiel et Scrum à distance		27h	
Sprint 3 (2 semaines)	*21/03/2024 - 03/04/2024*	**Client lourd:** Exigences souhaitables	Modes de jeu - Classique et Temps Limité augmenté (4 points)	13h	**144h** **31 pts**
			Clavardage - Envoyer des GIF (3 points)	9h	
			Personnalisation des sons (1 point)	6h	
			Clavardage - censures des mots inappropriés (1 point)	4h	
			Clavardage - Ajout de messages globaux (1 point)	4h	
			Clavardage - Ajout de messages aléatoires (1 point)	4h	
			Compte utilisateur - Réinitialiser le mot de passe (2 points)	6h	
			Compte utilisateur - Elo (3 points)	10h	
			Compte utilisateur et historique (1 point)	4h	
		Client léger: Exigences souhaitables	Modes de jeu - Classique et Temps Limité augmenté (4 points)	13h	
			Modes de jeu - secouer et crier sur la tablette (2 points)	8h	

			Personnalisation des sons (1 point)	4h	
			Clavardage - censures des mots inappropriés (1 point)	4h	
			Effet sonore observateur (1 point)	4h	
			Compte utilisateur - Réinitialiser le mot de passe (1 point)	6h	
			Clavardage - Notifications (1 point)	4h	
			Compte utilisateur - Elo (2 points)	10h	
			Evaluation des jeux (1 point)	4h	
		Déployer l'application sur AWS		4h	
		Rencontre hebdomadaire en présentiel et Scrum à distance		27h	
Sprint 4: Remise (2 semaines)	*3/04/2024 - 15/04/2024*	Présentation orale		30h	**144h**
		Corriger les bogues restants dans l'application		25h	
		Mettre à jour la documentation du sprint 1		30h	
		Rédiger le plan de tests et résultats des tests		20h	
		Rencontre hebdomadaire en présentiel et Scrum à distance		27h	
		Dernières révision du produit final (Application + Documentation + Tests)		12h	

5. Équipe de développement

L'équipe est constituée des 6 développeurs, tous étudiants en génie logiciel à Polytechnique, les voici: Lucas Bouchard, Ryan Lahbabi, Jeremy Rouillard, Zarine-Amy Ardekani-Djoneidi, Fatima-Zohra Oulaidi et Yin Zhiqin. Chaque membre par ses compétences et expériences passées apportera une expertise différente au projet et travaillera autant d'heures sur le projet pour des raisons d'équité :

Lucas Bouchard: Lucas apporte une expertise cruciale en développement de serveur et en gestion de bases de données. Sa responsabilité principale sera de concevoir l'architecture du serveur, la logique des algorithmes des différentes features et d'assurer son fonctionnement optimal. Son rôle est fondamental pour la stabilité et la performance du backend, garantissant ainsi que l'infrastructure sous-jacente supporte efficacement les applications front-end.

Zarine-Amy Ardekani-Djoneidi: Spécialiste du développement front-end, Zarine joue un rôle clé dans le développement des interfaces utilisateur pour les applications mobiles et de bureau. Sa capacité à créer des expériences utilisateur intuitives et esthétiquement agréables sera essentielle pour engager les utilisateurs. Zarine est doté d'une excellente expertise pour travailler sur la conception de features complexes et pour les programmer de façon efficace.

Fatima-Zohra Oulaidi : Experte en développement mobile, Fatima-Zohra orientera son expertise vers la création d'une application mobile épurée et efficace. Sa profonde connaissance des technologies mobiles sera un atout majeur pour assurer la réactivité, la fonctionnalité et l'optimisation des applications. Elle sera responsable de permettre expérience utilisateur fluide et intuitive, et garantira une performance maximale même sur des appareils aux capacités limitées.

Jeremy Rouillard: Expert en développement front-end, tant pour les applications mobiles que de bureau, Jeremy apporte également des compétences organisationnelles en tant que scrum master et gestionnaire de JIRA. Sa double compétence en développement et en gestion de projet sera un atout pour coordonner l'équipe et maintenir le projet sur la bonne voie. Nous l'avons choisie en tant que gestionnaire de projet car il est le plus expérimenté après avoir eu la chance de faire plusieurs stages dans l'industrie et surtout car il est détenteur de sa propre start-up.

Ryan Lahbabi: Spécialiste en développement front-end, il travaillera sur le développement des applications web et mobiles. Ryan sera aussi responsable du déploiement des applications. Grâce à sa certification d'AWS Cloud Practitioner, il possède une expertise dans la mise en œuvre et la gestion des solutions cloud. Cette compétence est importante pour assurer une infrastructure cloud robuste et sécurisée, facilitant ainsi la distribution efficace et fiable des applications.

Yin Zhiqin: Yin apporte une expertise essentielle en matière de conception UX, utilisant des outils comme Figma pour créer des designs intuitifs et attrayants. Sa spécialisation en développement front-end, couplée à ses compétences en UX, sera très utile pour développer des applications qui ne sont pas seulement fonctionnelles, mais aussi agréables à utiliser et intuitives. Il sera aussi responsable de la communication entre le serveur et le client en gérant les protocoles Http et WebSocket.

6. Entente contractuelle proposée

Selon les termes de notre appel d'offres, notre engagement est de développer une application ludique permettant de jouer au jeu de "Trouver les différences". Notre contrat est un contrat fixe et nous fournirons une version de cette application pour les ordinateurs (client lourd) et les appareils mobiles (client léger). Toutes nos fonctionnalités et spécifications seront détaillées dans le document de Spécification des Exigences du Système. Il est important de noter que notre équipe se réserve le droit de mettre en œuvre des exigences complémentaires, bien que différentes de celles prévues dans la planification du plan de projet, à condition que ces dernières soient spécifiées dans le SRS.

Notre équipe est composée de cinq développeurs et un gestionnaire de projet. Pour chaque membre, la charge de travail par personne est estimée à 12 heures par semaine pendant 14 semaines pour un total 168 heures pour la durée totale du projet. Cela représente donc 1008 heures*personnes. Nous estimons que de ce total 80% de ce temps sera dédié au développement du projet et 20% sera consacré à la gestion du projet et à la rédaction de sa documentation.

Notre équipe s'engage à respecter le calendrier établi, avec une date de livraison du produit final fixée au plus tard le 15 Avril 2024 à 23h59. A cette date, nous fournissons aussi une version mise à jour des documents d'architecture logicielle, le protocole de communication, le SRS, le plan de test et les résultats obtenus lors de ces tests.

Les honoraires demandés par notre équipe sont de 110$/heure pour les développeurs et 145$/heure pour le gestionnaire du projet. Par conséquent, le client doit s'engager à couvrir les coûts du projet qui vont s'élever à 116 760$. Le calcul est le suivant: 110$ x 840 heures + 145$ x 168 heures. Nous considérons que les coûts liés à l'infrastructure de l'application seront négligeables étant donné que nous utiliserons l'offre d'Amazon Web Services *AWS Free Tier*. Toutefois, dans le cas où une augmentation des capacités de stockage et de calcul des serveurs s'avèrent nécessaires pour garantir une expérience utilisateur à la fois satisfaisante et agréable avec l'application, il sera de la responsabilité du client de prendre en charge les coûts associés à l'utilisation du compte Amazon Web Services. Nous désirons également la mise à disposition par notre client d'un bureau équipé d'une connexion Wi-Fi haut débit (au moins 40 Mb/s), ce qui nous permettra de collaborer efficacement en équipe ou de travailler individuellement dans des conditions optimales pour le développement du projet.

En cas de modification du contrat, le client doit en informer l'équipe et accepter une extension du délai nécessaire qui sera jugé en fonction de l'ajustement requis.